(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平8－263243

(43)公開日　平成８年(1996)10月11日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　3/14	３１０		Ｇ０６Ｆ　3/14	３１０Ｃ	

審査請求　未請求　請求項の数３　ＦＤ　（全 22 頁）

(21)出願番号　特願平7－91911

(22)出願日　平成７年(1995)３月27日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　木村　康浩
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

(72)発明者　金井　達徳
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

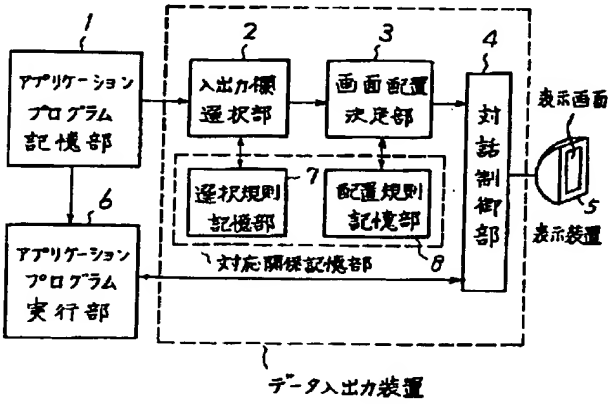
(74)代理人　弁理士　則近　憲佑

(54)【発明の名称】　データ入出力方法及び装置、及び計算機装置

(57)【要約】

【目的】　本発明はデータ入出力を伴うプログラミングの容易化及びプログラムの入出力装置やウィンドウシステムへの非依存を実現することを目的とする。

【構成】　アプリケーションプログラムから入出力対象となるデータの型定義をひとつあるいは複数受け取り、前記データ型の入出力に必要な画面上の入出力欄を選択する入出力欄選択部と、前記入出力欄選択部によって選択した複数の入出力欄の画面上の配置を決定する画面配置決定部と、前記画面配置決定部によって決定された配置にしたがって前記入出力欄選択部で選択した入出力欄を画面上に表示し、アプリケーションプログラムから渡された出力データを対応する出力欄に表示し、アプリケーションプログラムから要求されたデータの入力を対応する入力欄を通して実行する対話制御部から構成される。



1

【特許請求の範囲】

【請求項１】ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る表示画面を計算機の表示装置に出現させ、この表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うデータ入出力方法において、

アプリケーションプログラム実行時に、アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を該アプリケーションプログラムから抽出し、

この抽出されたデータの型に基づき、予め記憶されたデータの型と表示すべき画面の構成との対応関係を参照して、前記アプリケーションプログラムに対応する画面の構成を求め、

この求められた構成に従って表示画面を前記表示装置に出現させ、

この出現させた表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うことを特徴とするデータ入出力方法。

【請求項２】ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る画面を表示可能である表示装置を備える計算機に具備されるべきデータ入出力装置であって、

アプリケーションプログラムから、該アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を抽出する抽出手段と、

データの型と前記計算機の仕様に適合した表示すべき画面の構成との対応関係を複数のデータの型について記憶する記憶手段と、

前記抽出手段の抽出結果と前記記憶手段に記憶された対応関係に従って表示画面を前記表示装置に表示させる表示制御手段と、

この表示画面を介して入力されるデータの前記アプリケーションプログラムへの送信及び前記アプリケーションプログラムから出力されるデータの前記表示画面を介した提示の少なくとも一方を行う入出力制御手段とを備えたことを特徴とするデータ入出力装置。

【請求項３】ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る画面を表示可能である表示手段と、

前記アプリケーションプログラムから、該アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を抽出する抽出手段と、

データの型と前記表示手段に表示すべき画面の構成との対応関係を複数のデータの型について記憶する記憶手段と、

前記抽出手段の抽出結果と前記記憶手段に記憶された対応関係に従って表示画面を前記表示手段に表示させる表示制御手段と、

この表示画面を介して入力されるデータの前記アプリケーションプログラムへの送信及び前記アプリケーションプログラムから出力されるデータの前記表示画面を介した提示の少なくとも一方を行う入出力制御手段と、

この入出力制御手段により送信されたデータを用いた前記アプリケーションプログラムの実行及び入出力制御手段に対する前記アプリケーションプログラムの実行結果の出力の少なくとも一方を行う実行手段とを備えたことを特徴とする計算機装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、電子計算機においてアプリケーションプログラムを実行する際に、アプリケーションプログラムへユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る表示画面を計算機の表示装置に出現させ、この表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うデータ入出力方法、及び、このようなデータ入出力方法を行うために予め計算機に具備されているべきデータ入出力装置に関する。

【０００２】

【従来の技術】ビットマップディスプレイやキャラクタディスプレイあるいはグラフィックディスプレイ等の端末装置の画面を通してユーザとの入出力を行うプログラムを開発する場合、従来は、プログラマがまず使用する端末装置（表示装置を備える計算機）の持つ画面のサイズや表示できる文字や図形等の制限（以下「仕様」と呼ぶ）を念頭において画面設計を行った。次に使用する端末装置あるいはその端末装置上で動作しているウィンドウシステムの持つ制御コードやコマンドを組み合せて、その画面を表示するプログラムを作成する必要があった。このようなプログラムの作成は、画面上の文字や図形の配置を意識した細かくて煩雑なものであった。

【０００３】さらに、このようにして作成した画面表示のためのプログラムは特定の端末装置の機種（仕様）に依存しており、新しい端末装置を用いる場合には新たにプログラムを作成しなければならなかった。

【０００４】これは、一種類のアプリケーションプログラムに対して端末装置の機種数だけ異なる画面表示のためのプログラムを作成しなければならないことを意味する。したがって、プログラマは、アプリケーションプログラムの数をｎとし、端末装置の機種の数をｍとすると、ｎ×ｍの数だけ画面表示のためのプログラムを作成しなければならなかった。

【０００５】このような画面を用いて入出力を行うプログラムの作成を容易化する方法として、フォームジェネレータあるいはＧＵＩビルダと呼ばれるソフトウェアが

2

用いられることもある。これらのソフトウェアは、入出力に用いたい画面上の配置をプログラマが定義すると、その画面を表示するプログラムを自動生成するものである。このようなソフトウェアを用いることによって画面の配置の設計が一部楽にはなるが、画面の設計とアプリケーションプログラム本体の設計が分かれるため、一方を変更するとそれに合わせてもう一方も変更する必要があった。またこのようにして作成したプログラムは、やはり特定の端末装置の機種に依存したものとなっている。したがって、ｎ×ｍの数だけ画面表示のためのプログラムの作成が必要であるという問題は解決していなかった。

【０００６】

【発明が解決しようとする課題】本発明は、上述した問題点に鑑み、画面を介して入出力を行うアプリケーションプログラムの開発において、各アプリケーション毎に画面の詳細な設計を行い画面表示のためのプログラムの作成を行う作業を不要にすることによって、プログラムの作成を容易にすると共に、画面を介して入出力を行うアプリケーションプログラムを端末装置の機種に依存しない形で作成できるようにすることを目的とする。

【０００７】

【課題を解決するための手段】本発明（第１の発明）は、ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る表示画面を計算機の表示装置に出現させ、この表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うデータ入出力方法において、アプリケーションプログラム実行時に、アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を該アプリケーションプログラムから抽出し、この抽出されたデータの型に基づき、予め記憶されたデータの型と表示すべき画面の構成との対応関係を参照して、前記アプリケーションプログラムに対応する画面の構成を求め、この求められた構成に従って表示画面を前記表示装置に出現させ、この出現させた表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うことを特徴とする。

【０００８】本発明（第２の発明）は、ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る画面を表示可能である表示装置を備える計算機に具備されるべきデータ入出力装置であって、アプリケーションプログラムから、該アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を抽出する抽出手段と、データの型と前記計算機の仕様に適合した表示すべき画面の構成との対応関係を複数のデータの型について記憶する記憶手段と、前記抽出手段の抽出結果と前記記憶手段に記憶された対応関係に従って表示画面を前記表示装置に表示させる表示制御手段と、この表示画面を介して入力されるデータの前記アプリケーションプログラムへの送信及び前記アプリケーションプログラムから出力されるデータの前記表示画面を介した提示の少なくとも一方を行う入出力制御手段とを備えたことを特徴とする。

【０００９】本発明（第３の発明）に係る計算機装置は、ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る画面を表示可能である表示手段と、前記アプリケーションプログラムから、該アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を抽出する抽出手段と、データの型と前記表示手段に表示すべき画面の構成との対応関係を複数のデータの型について記憶する記憶手段と、前記抽出手段の抽出結果と前記記憶手段に記憶された対応関係に従って表示画面を前記表示手段に表示させる表示制御手段と、この表示画面を介して入力されるデータの前記アプリケーションプログラムへの送信及び前記アプリケーションプログラムから出力されるデータの前記表示画面を介した提示の少なくとも一方を行う入出力制御手段と、この入出力制御手段により送信されたデータを用いた前記アプリケーションプログラムの実行及び入出力制御手段に対する前記アプリケーションプログラムの実行結果の出力の少なくとも一方を行う実行手段とを備えたことを特徴とする。

【００１０】

【作用】本発明によれば、データの入出力を行うアプリケーションプログラムは、入出力を行いたいデータの型をデータ入出力装置に伝える。データ入出力装置は受け取ったデータ型のデータの入出力に必要な画面を合成し、その画面を用いてアプリケーションプログラムとユーザとの間のデータの入出力を行う。

【００１１】第１の発明は、アプリケーションプログラムの実行時に行われる動作手順についてのものである。第２の発明は、このような動作が行われるために、計算機のシステム側が備えるべき手段についてのものである。ここで言うデータ入出力装置とは、計算機のシステムにソフトウェアがインストールされることにより、計算機の内部に仮想的に上述した手段を備える装置に相当する機能が形成された場合には、その機能を実現する部分を指すものである。第３の発明は、上述した動作を行う計算機装置の構成についてのものである。

【００１２】このように、アプリケーションプログラム実行時に入出力を行いたいデータ型から自動的に画面を作成するため、アプリケーションプログラムのプログラマは、画面上の細かな配置を指定して画面表示のためのプログラムを予め作成しておく必要がなくなる。

【００１３】また、自動的に画面を作成する際に用いるために記憶されているデータ型と表示すべき画面の構成

との対応関係は、端末装置の機種に適合したものであるから、同じアプリケーションプログラムを複数の異なる端末装置で用いることが可能になり、アプリケーションプログラムが端末装置の機種に依存することをなくすことができる。

【００１４】さらに、本データ入出力装置はアプリケーションプログラム毎に異なるものを設ける必要はなく、アプリケーションプログラムの数をｎとし、端末装置の機種の数をｍとすると、本データ入力装置がｍ個存在さえすれば、どのアプリケーションプログラムをどの端末装置で実行する際にも、画面を介したデータの入出力が自動的に実現できる。

【００１５】

【実施例】本発明の一実施例を説明する。

【００１６】アプリケーションプログラムは入出力を行いたいデータのデータ型の定義を本実施例のデータ入出力装置に渡す。このとき用いるデータ型は、アプリケーションプログラムを記述しているプログラミング言語の持つデータ型を用いることも可能であるし、例えば「The Common Object Request Broker: Architecture andSpecification」(Object Management Group発行) に開示されているCORBA/IDLのような特定のプログラミング言語に依存しないデータ型を用いることもできる。

【００１７】ここではCORBA/IDL を用いる場合の実施例を示すが、他のデータ型を用いる場合も本実施例と同様に扱うことが可能である。

【００１８】データ入出力装置の入出力欄選択部２（後述する）にデータ型の定義を与えるためには、入出力欄選択部２によって解釈可能なデータとしてデータ型を表現する必要がある。これは例えば図２及び図３（図２の続き）に示すようなC 言語の構造体によってCORBA/IDLのデータ型を表現するデータ型によって実現することが可能である。図２及び図３において typecode という名前を持つ構造体が、データ型を表現するデータ型である。 typecode 構造体中の kind フィールドが、int か char か構造体かといった型の種類を表現し、構造体は配列のような複合型の場合はそれぞれの型の定義に必要な情報を例えば構造体なら t_struct というフィールドに持っている。

【００１９】CORBA/IDL には、データ型を表現するメタなデータ型として TypeCode と呼ぶ型が定義されている。図２及び図３における typecode 型の構造体は、CORBA/IDL のデータ型における TypeCode をC 言語のデータ型に変換したものである。

【００２０】以下、本実施例では、アプリケーションプログラムから入出力欄選択部２に渡されるデータの型定義は、CORBA/IDL のTypeCode型のデータとしてで渡されるものとして説明する。

【００２１】図４及び図５（図４の続き）に本実施例のデータ入出力装置を用いるアプリケーションプログラムの例を示す。この例のアプリケーションプログラムは、CORBA/IDL のデータ型とC 言語の文法を持つオブジェクト指向言語で記述されている。

【００２２】図示したプログラム中でuim は本発明のデータ入出力装置に対応するオブジェクトである。uim.input 命令は第3 の引数に渡したデータ型の入力欄を作成することをuim オブジェクトに指示している。これによって、uim オブジェクトによって実現されている本実施例の入出力欄選択部２に入力したいデータのデータ型定義が渡される。同様にuim.output命令は第3 の引数に渡したデータ型の出力欄を作成することをuim オブジェクトに指示している。uim.event 命令は、ボタンのように、その欄に対応付けられた動作を起動する欄を生成することを指示する命令である。起動する動作は、第3 引数に関数へのポインタとして与える。

【００２３】uim.dialogは、uim.input 命令やuim.output命令、uim.event 命令、さらに再帰的にuim.dialog命令で作成される入出力欄の間に階層関係定義するために用いる。つまり、input やoutput、event 、dialog命令の第2 引数には、その階層上の親となるdialogを指定でき、これによって階層構造を定義できる。dialogはinput やoutput、event 、dialogを含んだ画面上の意味のあるまとまりを示すものである。uim.input で作成した入力欄からは、入力命令であるget 命令で値を入力することができる。uim.outputで作成した出力欄には、出力命令であるput 命令で値を出力することができる。

【００２４】また本実施例では、入出力欄の画面への表示にMIT で開発されたX ウィンドウシステムおよびその標準的なツールキットであるOSF(Open Software Foundation) のMotif ツールキットを用いる場合について説明するが、Microsoft 社のWindows を用いる場合も同様の方式で実施可能である。

【００２５】図１は本発明の構成を説明する説明図である。この図において、１はアプリケーションプログラム記憶部、６はアプリケーションプログラム実行部、２はアプリケーションプログラム（１）から入出力を行いたいデータ型の定義を一つまたは複数受け取り、受け取ったデータ型に対応してその入出力に必要な表示画面上の入出力欄の種類を選択する入手力欄選択部、３は２で選択した複数の入出力欄の画面上での配置を決定する画面配置決定部、４は３で決定された配置に従って２で選択した入出力欄を画面上に表示し、アプリケーションプログラム（６）から渡された出力データを対応する出力欄に表示し、アプリケーションプログラム（６）から要求されたデータの入力を対応する入力欄を通して実行する対話制御部、５は入出力欄を含む画面が表示される表示装置である。

【００２６】なお、入出力欄選択部２の処理の際は選択規則記憶部７に記憶された情報を参照し、画面配置決定部３の処理の際は配置規則記憶部８に記憶された情報を

参照する。記憶部7及び8は全体として、複数のデータ型について、データ型とこの計算機の仕様に適合した表示すべき画面の構成との対応関係の情報を記憶していることになる。また、入出力欄選択部2の前段は、アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータのデータ型の定義を抽出する手段を含んでおり、入出力欄選択部2の後段及び画面配置決定部3は全体として、前記の抽出結果に基づき、前記の記憶された対応関係の情報を参照して、記憶部1に記憶され実行部6で実行されつつあるアプリケーションプログラムに対応する画面の構成を求める手段を含んでいる。

【0027】なお、アプリケーションプログラムが入出力を行うデータが一つである場合は、画面配置決定部3がなくても、表示すべき画面の構成を求めることは可能である。対話制御部4は、この求められた構成に従って表示画面を出現させる手段と、この表示画面を介して入力されるデータをアプリケーションプログラムへの送信したりアプリケーションプログラムから出力されるデータを表示画面を介して提示したりする手段とを含んでいる。

【0028】そして、上記2～4及び7、8は、データ入出力装置を構成しており、これに1、5及び6を加えた全体が、計算機装置を構成している。

【0029】まずアプリケーションプログラムには、入出力対象となるデータの型定義がそれぞれTypeCodeで表現され記載されている。そして、入出力欄選択部2は、記憶部1に記憶されたアプリケーションプログラムから、このTypeCodeの値を検出する。

【0030】入出力欄選択部2は、検出されたTypeCodeの内容を参照し、あらかじめ定められた選択規則記憶部7に記憶された選択規則に則って、各データの型に対応した入出力欄の情報を決定し、画面配置決定部3に渡す。入出力欄選択部2の決定する入出力欄の情報は、以下の要素からなる。

【0031】(1) 表示される入出力欄の構成。

【0032】本発明をXウィンドウシステムを用いて実施する場合、アプリケーションプログラムが指定したデータ型に対応して実際に入出力を行うために、画面上にウィジェットと呼ぶ入出力欄を構成する部品を表示する。ウィジェットには、トグルボタンを表示する ToggleButton ウィジェット、文字列を表示する Text ウィジェット、複数のウィジェットを指定した並び方で表示する RowColumnウィジェットなどがある。具体的には図6に示すようなウィジェットの階層構造によって表示される入出力欄の構成を表す。たとえば、図4及び図5に示すプログラムにおいて「ハンバーガー型」のデータ型の入力命令に対応して作成されるウィジェットの階層構造は図6のようになる。

【0033】(2) 生成した入出力欄が実行時に必要とする手続き。

【0034】例えば入出力欄がScrollbar ウィジェットを用いている場合には、Scrollbarが操作された時にスクロール動作を行なう手続きをScrollbar に登録する必要がある。したがって「スクロール動作を行なう手続きをScrollbar ウィジェットに登録すること」が入出力欄生成後に実行すべき手続きの一つとなる。以下の説明では入出力欄生成後に実行すべきこのような手続きのことを「後処理手続き」と呼ぶことにする。

【0035】(3) アプリケーションプログラムからの入力命令によって作成された入出力欄に対しては、その欄に対するユーザからの入力データを読みとってアプリケーションプログラムに渡すのに必要な手続き。

【0036】(4) アプリケーションプログラムからの出力命令に対して作成された入出力欄に対しては、アプリケーションプログラムから受けとったデータを画面上に表示するための手続き。

【0037】以下、CORBA/IDL の主なデータ型に対応して入出力欄選択部2が決定する入出力欄の構成，後処理手続きおよび入出力命令に対する手続きの選択規則を示す。ここに示していないCORBA/IDL のデータ型に関しても同様の規則を適用することが可能である。

1) データがboolean 型の場合

この型の入出力欄としてはToggleButtonウィジェットを用い、ToggleButtonのON,OFFで真理値のTrueとFalse を表現する。したがって入出力欄の構成としては図7のようにToggleButtonウィジェット一つとなる。後処理手続きは特にない。またデータの出力はToggleButtonの状態を適切に設定することであるから、データ出力手続きは図9のようになる。同様にデータの入力のためにはToggleButtonの状態を調べる必要があるから、データ入力手続きは図10のようになる。

2) データがshort, long, hyper, float, double 型の場合

これらの数値型の入出力欄としてはTextウィジェットを用いる。したがって入出力欄の構成は図8のようにTextウィジェット一つとなる。数値型の入出力欄の後処理手続きは以下のようになる。

・入力可能な最大文字数および表示桁数を、その型で扱うことのできる全ての値を表現するのに必要かつ十分な文字数に設定する。例えばshort 型が扱うことのできる範囲は-32768から32767 までであるから、short 型の全ての値を表現するのに必要かつ十分な文字数は6 である。したがってshort 型の入出力欄では入力可能な最大文字数および表示桁数を6 に設定する。

・その型を表現するのに必要かつ十分な文字のみを正しい入力として受け付けるようにする。例えばshort 型の場合0,1,2,3,4,5,6,7,8,9,- の11文字のみを入力として受け付けるようにする。

【0038】したがって後処理手続きとして二つがある。一つはTextの幅を適切に設定するためのもので、図

１１のようになる。なお図１１における「その型の表現に必要十分な文字数n を求める」方法としては、例えば型と文字数n とからなる表をあらかじめ作成しておきそれを引くという方法を用いることができる。もう一つは妥当な文字のみを入力として受け付けるようにするためのもので、そのためにはユーザからの入力があった時に図１２に示されたような入力の妥当性をチェックする手続きを実行する必要がある。Textウィジェットはユーザからの入力があった時に特定の手続きを実行するように設定することが可能になっている。従って二つ目の後処理手続きは図１３のようになる。

【００３９】最後に入出力命令に対する手続きとしては、入力命令に対してはText内に入力された文字列を読みとり、数値に変換した後にアプリケーションプログラムに渡す。出力命令に対しては、受けとったデータを文字列に変換し、その文字列をText上に表示する。したがってデータ入力手続きは図１４のように、　データ出力手続きは図１５のようになる。

3)　データがenum型の場合

この型の入出力欄としては、複数個のToggleButtonウィジェットとRowColumnウィジェットから構成されるRadioBoxを用いる。RadioBoxでは複数のToggleButtonのうち一つだけが押された状態になり得る。押された状態のToggleButtonがデータの値を表現する。よって入出力欄の構成としては、図１６のようにRowColumn ウィジェットとそれを親ウィジェットとする列挙子の数に等しいToggleButtonウィジェットという構成になる。

【００４０】次に後処理手続きは、enum型の入出力欄として用いるためには、生成されたToggleButtonが前述のようなRadioBoxとして動作するように設定する。しかしRowColumn ウィジェットのリソース(動作方式を変更するパラメタ) の中には、子ウィジェットとして生成されたToggleButtonをRadioBoxとして動作させるのに必要な設定を自動的に行なうか否かを決めるリソースが存在する。そしてMotif にはこのリソースを設定した状態でRowColumn を生成する手続きが存在する。したがってRowColumn を生成するのにこの生成手続きを用いれば、後処理手続きは特になくなる。

【００４１】最後に入出力命令に対する手続きについては、入力命令に対しては各ToggleButtonの状態(押されているか否か) を調べ、押されている状態のToggleButtonに対応する値をアプリケーションプログラムに渡す。従って手続きは図１８のようになる。一方出力命令に対しては、受けとったデータの値に対応するToggleButtonを押させた状態に設定すれば良い。従って手続きは図１９のようになる。

4)　データがstring型の場合

string型の入出力欄は、ScrolledWindowウィジェットとTextウィジェットからなるScrolledTextを用いる。従って構成は図１７のようにScrolledWindowとその子供のTextという構成になる。このような構成の画面がScrolledTextとして動作するためにはScrolledWindowウィジェットの中のScrollbar が操作された時にTextウィジェットの表示内容がスクロールするような設定をする必要がある。しかしMotif にはScrolledWindowウィジェットおよびTextウィジェットを生成した後ScrolledTextとして動作するために必要な設定を自動的に行なう手続きが提供されている。従って画面の生成にそのような手続きを用いることにすれば、後処理手続きは特にない。入力命令に対してはユーザからのTextウィジェットへの入力を読みとる手続きを実行し、その値をアプリケーションプログラムに渡す。出力命令に対しては受けとったデータの値をTextウィジェット上に表示する手続きを実行する。従って入力命令に対する手続きは図２０のように、出力命令に対する手続きは図２１のようになる。

5)　データがstruct型の場合

struct型の入出力欄としては、structの各メンバに対応する入出力欄をRowColumn ウィジェットの中に入れて一つにまとめたものを用いる。従って構成は図２２のようにRowColumn の下に各メンバにあたるウィジェット階層がつながる形となる。図２２において一点破線で囲まれた領域は各メンバの型に対応する入出力欄のウィジェット階層である。この構成は図２３に示されるような手順によって決定される。すなわち、まずTypeCodeの中の1番目のメンバの型に関する情報を調べ、その型のための入出力欄の構成を決定し、決定した構成を1 番目のメンバ用の入出力欄とする。これをsturctの各メンバについて繰り返し、得られた各メンバのウィジェットツリーをRowColumn の下につないで得られる図２２のようなウィジェットツリーをこのstruct型全体の入出力欄構成とする。

【００４２】次に後処理手続きであるが、これも入出力欄の構成の決定と同様の手順となる。すなわち一つのメンバについてそのメンバの型の入出力欄を生成した際の後処理手続きを決定し、これをsturct型の各メンバについて繰り返す。その結果挙げられた手続き全体がこのstruct型の後処理手続きとなる。以上をフローチャートにすると図２４のようになる。

【００４３】最後にアプリケーションプログラムからの入出力命令に対して実行される手続きであるが、入力命令に対しては、まず1 番目のメンバについてそのメンバの型に応じた入力手続きを実行しそのメンバの値を得る。これを各メンバについて繰り返し各メンバの値を得る。全てのメンバの値が得られると元のデータの値が決まるので、その値をアプリケーションプログラムに渡す。出力命令に対しては、受けとった値を調べて1 番目のメンバの値を得る。次にそのメンバの型に応じた出力手順を実行して1 番目のメンバの値を画面上に表示する。これを全てのメンバについて繰り返し実行することによりもとのstruct型データ全体が画面に表示される。

従って入力命令に対する手続きは図２５のように、出力命令に対する手続きは図２６のようになる。

6) データがunion 型の場合

一つの記憶領域にその時々に応じてさまざまな型のデータを格納できるのがunion 型である。記憶領域に格納できるデータ型の候補をそのunion 型のメンバという。CORBA/IDL のunion 型では記憶領域の中に現在のデータの値と、現在格納されているデータに対応するメンバの識別子、すなわち現在格納されているデータの型を識別する情報が保存される。従ってunion 型の入出力欄にはメンバの識別子用の欄と値用の欄が必要になるため、図２７のようにメンバを識別するRadioBox１０と値用の欄を表示する領域１１とからなるものを用いる。後者にはRadioBoxによって指定されるメンバの型に応じた入出力欄が表示される。つまり、この領域１１の中にはそれぞれのメンバの型に対応する入出力欄が非表示状態で生成されており、RadioBoxの中の押された状態のToggleButtonに対応するメンバ用入出力欄だけが表示状態とされて領域１１の中に表示される。このようなstruct型入出力欄のウィジェット階層は図２８のようになる。この図において１２はRowColumn ウィジェットおよびその子供のToggleButtonウィジェットからなるRadioBoxであり、１３はunion の各メンバの型に応じた入出力欄のウィジェットツリーである。各メンバ用入出力欄のウィジェットツリーは全てFormウィジェットを親ウィジェットとしており、このFormウィジェットが図２７の領域１１にあたる。なお図２８のウィジェットツリーのうち各メンバ用の入出力欄については、struct型の場合と同様に、TypeCodeのメンバの型に関する情報を調べその型のための入出力欄を決める、という手順を各メンバについて繰り返すことによって得られる。したがって図２８のウィジェットツリー全体を決定する手順は図２９のようになる。

【００４４】次にunion 型の後処理手続きとしては以下の二つがある。

・union の各メンバについて、そのメンバの型に応じた後処理手続きを実行する手続き。

・RatioBox内のToggleButtonが押された時にそのToggleButtonに対応するメンバの入出力欄が入出力欄表示領域に表示されるように設定する手続き。

【００４５】前者に関してはstruct型の場合と同様に、一つのメンバについてそのメンバの型の入出力欄を生成した際の後処理手続きを決定し、これをunion 型の各メンバについて繰り返す。その結果挙げられた手続き全体がこの一つ目の後処理手続きとなる。後者については、ToggleButtonが押された時に図３０に示されたような手続きを実行する設定をRadioBox内の各ToggleButtonに対して行なう必要があるから、実行すべき手続きは図３１のようになる。

【００４６】最後にアプリケーションプログラムからの入出力命令に対応するための手続きとしては、入力命令に対してはまずRadioBox内のToggleButtonの状態を調べることにより現在欄に入力されているデータがどのメンバにあたるかを知る。次に該当するメンバの型に対応する入力手続きを実行して、表示状態になっているそのメンバ用入出力欄に入力されている値を読みとる。以上によりメンバの識別子およびデータの値が得られたので元のunion 型データの値が定まる。よってその値をアプリケーションプログラムに渡す。一方出力命令に対しては、まず受けとったデータ内のメンバ識別子を調べ、そのメンバに対応するToggleButtonが押された状態にRadioBoxを設定する。するとそのメンバに対応する入出力欄が表示状態となるので、メンバの型に対応する出力手続きを実行して値を表示状態となっている入出力欄に表示する。以上より入力命令に対する手続きは図３２のように、出力命令に対する手続きは図３３のようになる。

7) データがarray 型の場合

array 型の入出力欄としては、その要素の型に対応する入出力欄を複数並べたものを用いる。その際要素の個数だけ並べることにすると、要素の個数が多い場合には欄が画面に収まり切らないなど不都合が生じる。そこで要素の入出力欄を並べる個数に上限を設け、ScrollBar を用いることにより表示される要素の範囲を変更できるようにする。したがって入出力欄の構成は図３４のようになる。

【００４７】ところでこのような入出力欄を用いた場合、array の要素数が前述の上限を超えると要素数と要素を表示する欄の個数が一致しなくなる。そのためarray 型ではアプリケーションプログラムからの入力命令によって渡されたデータやユーザからの画面への入力を保存しておく記憶領域が必要になる。またScrollBar が操作された時に、ユーザからの入力を前記記憶領域に書き出した上で、個々の要素用入出力欄の表示を表示範囲の変更に合うように書き直すように設定することが必要となる。したがってunion 型の後処理手続きとしては次の三つがあげられる。

・array の要素の型に対応した後処理手続きをすべての要素用入出力欄に対して実行する手続き。

・生成された入出力欄が用いる前記記憶領域を動的に割り当てる手続き。

・ScrollBar の操作によるスクロールに対応するために、ScrollBar が操作された時に図３５のような手続きが実行されるように設定する手続き。

【００４８】最後にアプリケーションプログラムからの入出力命令に対応するための手続きであるが、入力命令に対しては、まず要素の型に応じた入力手続きを実行して要素用入出力欄に入力されている値を読みとり、その値を前記記憶領域の対応する要素の場所に書き込む。これをすべての要素用入出力欄について繰り返した後、記憶領域に保存されている値をアプリケーションプログラムに渡す。一方出力命令に対しては、まず受けとった値

を記憶領域に書き込む。その後要素の型に応じた出力手続きを実行して、各々の要素用入出力欄に対応する要素の値を表示する。以上より入力命令に対する手続きは図３６のように、出力命令に対する手続きは図３７のようになる。

【００４９】入出力欄選択部２は、以上述べたような選択規則（選択規則記憶部７に記憶されている）に従い、検出したTypeCodeから入出力欄の構成，後処理手続きおよび入出力命令に対する手続きを、アプリケーションプログラムの入出力対象となるデータ毎に決定する。すなわち、検出したデータ型が上記1)～7)のいずれに相当するかに基づき、データ型に対応する選択規則を参照し、前述した入出力欄の情報を各データ毎に求める。そして決定された入出力欄の構成，後処理手続きおよび入出力命令に対する手続きを画面配置決定部３に渡す。

【００５０】なお、ここでの選択規則は、データ型とこの計算機の仕様に適合した表示すべき画面の部分構成（さらに後処理手続き及び入出力命令に対する手続き）との対応関係を、各データ型毎に表したものとなっている。この例では、計算機の仕様がX ウィンドウシステムである場合の、選択規則の例を示したが、別の仕様を持つ計算機に対しては、上記各データ型に対する規則は別のものとなる。そして、各仕様の計算機毎に、選択規則が設定される各データ型は共通にし、選択規則本体はその計算機の仕様に適合したものを記憶しておけば、アプリケーションプログラムは特定の計算機の仕様に依存せず、本データ入出力装置を備えるどの計算機でもそのままの形で実行可能となる。

【００５１】画面配置決定部３は、入出力欄選択部２から受けとった個々の入出力欄の構成（画面の部分構成）に基づき、画面上の入出力欄の全体的な構成，配置を決定する。画面配置決定部３は、まず、入出力対象のデータが複数ある場合に、複数の入出力欄を画面上にいかに配置するかを決定する。本例では、簡単な画面配置法として、複数の入出力欄を縦に（あるいは横に）一列に並べるという配置法を用いる。これは個々の入出力欄をRowColumn ウィジェットの中に入れることにより実現できる。この場合、全体的な画面構成としては図３８のような個々の入出力欄のウィジェットツリーをRowColumn の下につないだ形になる。

【００５２】画面配置決定部３は、さらに、上記のように複数の入出力欄を一列に並べた場合の入出力欄全体の大きさを求め、この入出力欄全体の大きさが表示装置５の画面の大きさを超える場合には、入出力全体へのアクセスを可能とする手段の使用を決定する。具体的には全体をScrolledWindowの中に入れて画面内に収まるようにする。例えば図３８のような画面構成の入出力欄が多き過ぎてScrolledWindowを用いた場合、その画面構成は図３９のようになる。

【００５３】このような配置法は、配置規則記憶部８に記憶された配置規則を参照することにより実現する。上記の選択規則は、各データ型に対応して計算機の仕様に適合した表示すべき画面の部分構成を示しており、実際のアプリケーションプログラムに含まれる複数のデータのそれぞれに対して画面の部分構成を求めるためのものであるのに対し、配置規則は、これら各データ毎に求められた複数の画面の部分構成を基に、画面全体の構成を求めるためのものである。

【００５４】なお、ここで例示した配置規則は、計算機の仕様がX ウィンドウシステムである場合の例であるが、別の仕様を持つ計算機は、別の配置規則を記憶する。例えば、画面のスクロール機能を有さない計算機の場合、複数の入出力欄を一列に並べ、並べたときの入出力欄全体の大きさが画面の大きさを超えるならば、入出力欄を複数の頁に割り付け、頁切り換え機能によって入出力欄全体をユーザに見せるようにする。

【００５５】画面配置決定部３はこのようにして決定された入出力欄の全体的な画面構成、および入出力欄選択部２が決定した後処理手続きと入出力命令に対する手続きを対話制御部４に渡す。

【００５６】図４０は、図４及び図５のアプリケーションプログラムに対して画面配置決定部３で作成された画面構成のウィジェットの階層構造を示している。さらに、図４１はそれに対応して表示される画面の例を示している。

【００５７】対話制御部４は、画面配置決定部３から受けとった入出力欄全体の画面構成に基づきX ウィンドウシステムの各種手続きを用いてを描画を行ない、その後で入出力欄選択部２が決定した後処理手続きを実行することにより入出力欄を画面上に表示する。

【００５８】また対話制御部４は、アプリケーションプログラムからの入出力命令に対する処理を行なう。すなわち入力命令に対しては、入出力欄選択部２が決定した入力命令に対する手続きを実行して、ユーザからの入力を読みとりアプリケーションプログラムに渡す。また出力命令に対しては、同じく入出力欄選択部２が決定した出力命令に対する手続きを実行することにより受けとったデータを画面に表示する。

【００５９】以上述べた手続きを、以下のデータ型に適用した場合の動作を説明する。

```
enum example__enum {foo,bar,baz}; …(1)
struct example__data__type {
  short  member1;
  string member2;
  example __enum member3;
};       …(2)
```

(1) はfoo,bar,baz という三つの識別子を持つexample __enumというenum型を定義しており、(2) はshort,string及びexample __enum型の三つのメンバを持つexample __data__typeというstruct型を定義している。以下では

15

このexample __data__type型のデータの入出力を行なう場合について説明する。

【００６０】まずアプリケーションプログラムはexample __data__type型の定義をTypeCode型のデータに変換し入出力欄選択部２に渡す。なお、上記の実施例では、アプリケーションプログラムが入出力を行うデータ型の定義をアプリケーションプログラムから抽出する動作は、入出力欄選択部２がアプリケーションプログラム記憶部１を参照して行うと説明したが、アプリケーションプログラム実行部６が抽出動作を行い、抽出結果を入出力欄選択部２へ渡しても良い。すなわち、アプリケーションプログラムからのデータ型の定義の抽出は、データ入出力装置側で行なっても良いし、アプリケーションプログラム実行部で行なって抽出結果をデータ入出力装置へ渡しても良く、本計算機装置内のどこに上記抽出手段があっても、本発明の範囲を逸脱しない。受けとったTypeCode型データに基づき入出力欄選択部２は以下の事項を決定する。

・入出力欄を構成するウィジェットの階層構造。
・ウィジェット生成後に実行すべき後処理手続き。
・アプリケーションプログラムからの入力命令に対して、画面に入力されたデータをアプリケーションプログラムに渡すために実行すべきデータ入力用手続き。
・アプリケーションプログラムからの出力命令に対して、アプリケーションプログラムから渡されたデータを画面に表示するために実行すべきデータ出力用手続き。

【００６１】まずウィジェットの階層構造を決める。受けとったTypeCodeから、入出力欄選択部２は入出力対象のデータがstruct型であることを知る。そこで入出力欄選択部２は、まず各メンバの型を調べてそのメンバ用の入出力欄を決める。member1というメンバはshort 型であるから、その入出力欄としてTextウィジェットを用いる。member2 というメンバはstring型であるから、その入出力欄としてはScrollWindowウィジェットとTextウィジェットからなるScrolledTextを用いる。member3 というメンバはexample __enum型であり、このexample __enum型は三つの識別子を持つenum型であるから、その入出力欄としてはRowColumn ウィジェットと３個のToggleButtonウィジェットからなるRadioBoxを用いる。各メンバ用入出力欄のウィジェット構成が決まったら、それらをRowColumn ウィジェットの中に収めたものをstruct全体の入出力欄とする。したがって入出力欄を構成するウィジェットの階層構造は図４２のように決定される。

【００６２】次に入出力欄決定部２は、後処理手続き、データ入力用手続き、データ出力用手続きを決め、ウィジェットの階層構造とあわせて画面配置決定部３へ渡す。画面配置決定部３はこの中のウィジェット階層構造に基づき、画面全体の構成を決定する。今表示に用いられる画面が小さく、ウィジェット階層構造に基づき表示される入出力欄全体が画面内に収まらない大きさになったとする。この場合画面配置決定部３は欄全体へのアクセスを可能にするためにScrollWindowの使用を決定する。したがって画面全体の構成として図４３のようなウィジェット階層構造を決定する。そしてこのウィジェット階層構造及び入出力欄決定部２から渡された後処理手続き、データ入力用手続き、データ出力用手続きを対話制御部４に渡す。

16

【００６３】対話制御部４はX ウィンドウシステムの各種手続きを呼び出すことにより図４３のウィジェット階層構造に従った画面を生成する。その後に後処理手続きを実行するわけであるが、この場合実行すべき後処理手続きはない。以上により図４４のような画面が表示される。また対話制御部４はアプリケーションプログラムからの入力命令に対しては入力命令用手続きを実行し得られたデータを渡し、出力命令に対しては出力命令用手続きを実行してデータを表示する。

【００６４】

【発明の効果】以上説明したように、本発明によれば、データの入出力を行うアプリケーションプログラムの開発を容易にすると共に、データ入出力を行うアプリケーションプログラムの特定の入出力装置やウィンドウシステム等への依存性を無くすことができる。

【図面の簡単な説明】

【図１】　本実施例の全体構成を表す図。
【図２】　データ型の定義の表現例を示す図。
【図３】　データ型の定義の表現例を示す図。
【図４】　データの入出力を行うアプリケーションプログラムの例を示す図。
【図５】　データの入出力を行うアプリケーションプログラムの例を示す図。
【図６】　表示される入出力欄の構成をウィジェットの階層構造によって表した例を示す図。
【図７】　データがboolean 型の場合の入出力欄の構成例を示す図。
【図８】　データがshort, long, hyper, float, double 型の場合の入出力欄の構成例を示す図。
【図９】　データがboolean 型の場合の入力命令に対する手続き例を示す図。
【図１０】　データがboolean 型の場合の出力命令に対する手続き例を示す図。
【図１１】　データがshort, long, hyper, float, double 型の場合の第１の後処理手続きの例を示す図。
【図１２】　データがshort, long, hyper, float, double 型の場合の第２の後処理手続きの例を示す図。
【図１３】　データがshort, long, hyper, float, double 型の場合の第２の後処理手続きの例を示す図。
【図１４】　データがshort, long, hyper, float, double 型の場合の入力命令に対する手続き例を示す図。
【図１５】　データがshort, long, hyper, float, double 型の場合の出力命令に対する手続き例を示す図。

17

【図１６】　データがenum型の場合の入出力欄の構成例を示す図。
【図１７】　データがstring型の場合の入出力欄の構成例を示す図。
【図１８】　データがenum型の場合の入力命令に対する手続き例を示す図。
【図１９】　データがenum型の場合の出力命令に対する手続き例を示す図。
【図２０】　データがstring型の場合の入力命令に対する手続き例を示す図。
【図２１】　データがstring型の場合の出力命令に対する手続き例を示す図。
【図２２】　データがstruct型の場合の入出力欄の構成例を示す図。
【図２３】　データがstruct型の場合の入出力欄の構成の決定法の例を示す図。
【図２４】　データがstruct型の場合の後処理手続きの例を示す図。
【図２５】　データがstruct型の場合の入力命令に対する手続き例を示す図。
【図２６】　データがstruct型の場合の出力命令に対する手続き例を示す図。
【図２７】　データがunion 型の場合の入出力欄の構成例を示す図。
【図２８】　データがunion 型の場合の入出力欄の構成例を示す図。
【図２９】　データがunion 型の場合の入出力欄の構成の決定法の例を示す図。
【図３０】　データがunion 型の場合の第２の後処理手続きの例を示す図。
【図３１】　データがunion 型の場合の第２の後処理手続きの例を示す図。
【図３２】　データがunion 型の場合の入力命令に対する手続き例を示す図。

18

＊【図３３】　データがunion 型の場合の出力命令に対する手続き例を示す図。
【図３４】　データがarray 型の場合の入出力欄の構成例を示す図。
【図３５】　データがarray 型の場合の第３の後処理手続きの例を示す図。
【図３６】　データがarray 型の場合の入力命令に対する手続き例を示す図。
【図３７】　データがarray 型の場合の出力命令に対する手続き例を示す図。
【図３８】　画面配置決定部で決定される全体的な画面構成の例を示す図。
【図３９】　画面配置決定部で決定される全体的な画面構成の例を示す図。
【図４０】　画面配置決定部で決定される全体的な画面構成の例を示す図。
【図４１】　図４０に対応して表示される画面の例を示す図。
【図４２】　入出力欄の構成を表すウィジェットの階層構造の例を示す図。
【図４３】　画面全体の構成を表すウィジェットの階層構造の例を示す図。
【図４４】　図４３に対応して表示される画面の例を示す図。

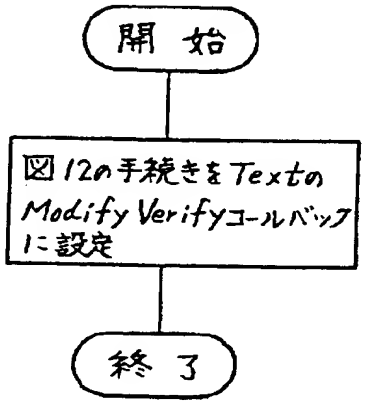
【符号の説明】
１…アプリケーションプログラム記憶部
２…入出力欄選択部
３…画面配置決定部
４…対話制御部
５…表示装置
６…アプリケーションプログラム実行部
７…選択規則記憶部
８…配置規則記憶部

＊

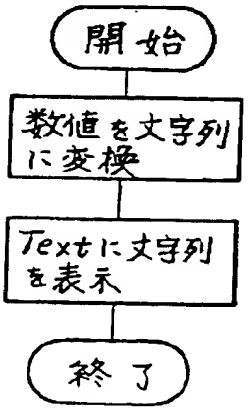
【図７】

【図８】

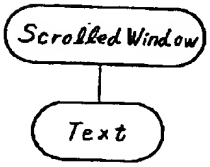
【図１３】



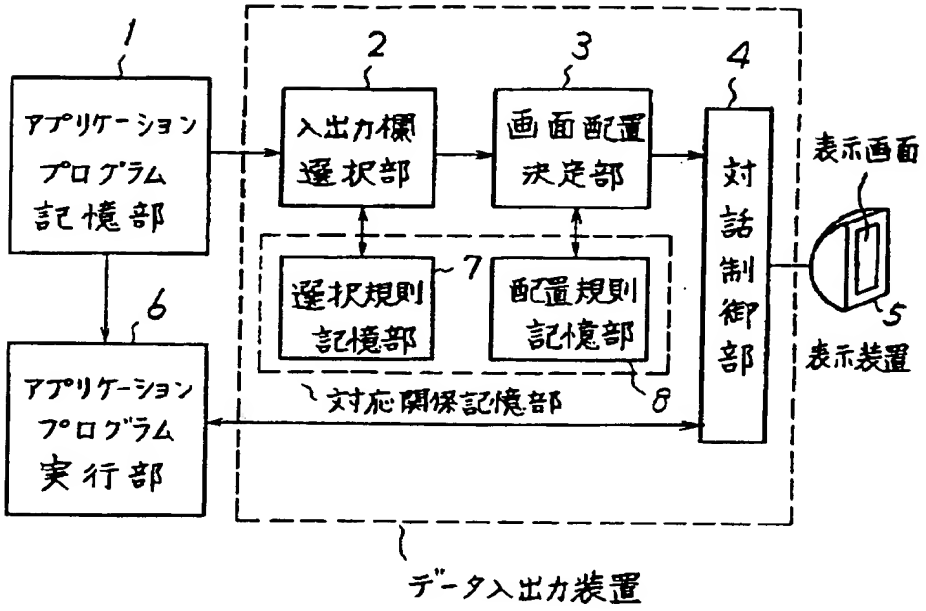
【図１５】



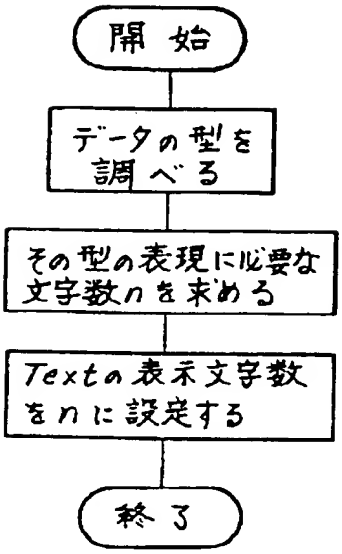
【図１７】



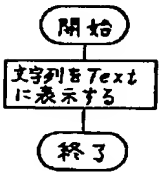
【図1】



【図11】



【図21】



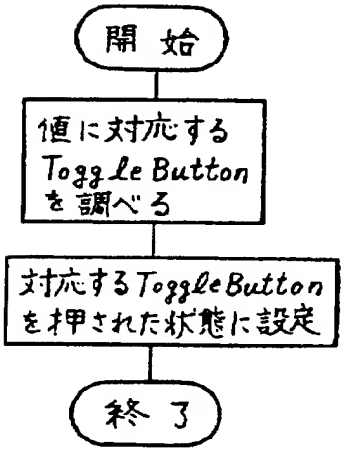
【図3】

```
typedef struct {
        unsigned long   maxlength;
        typecode        *type;
    } tc_sequence_struct;

typedef struct {
        unsigned long   length;
        typecode        *type;
    } tc_array_struct;

typedef enum tc_kind {
        tc_void, tc_boolean, tc_char, tc_octet, tc_short, tc_long,
        tc_ushort, tc_ulong, tc_float, tc_double, tc_struct, tc_union,
        tc_enum, tc_string, tc_sequence, tc_array, tc_any, tc_typecode,
      } tc_kind;

typedef struct typecode {
        enum tc_kind    kind;
        union {
            tc_struct_struct    t_struct;
            tc_union_struct     t_union;
            tc_enum_struct      t_enum;
            tc_string_struct    t_string;
            tc_sequence_struct  t_sequence;
            tc_array_struct     t_array;
        }                       opt;
    } typecode;
```

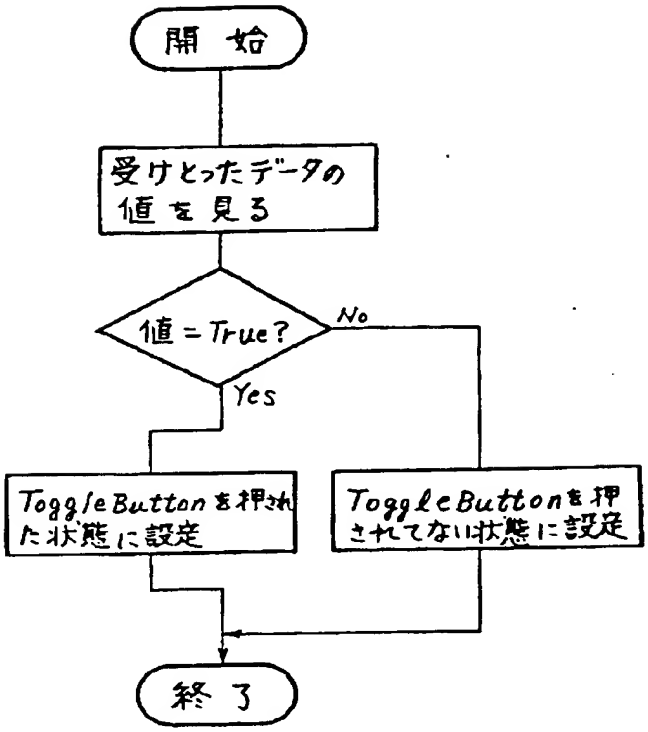
【図２】

```
typedef struct {
        char        *name;
        typecode    *type;
    }   tc_struct_member;

typedef struct {
        char                *name;
        unsigned long       memnum;
        tc_struct_member    *member;
    } tc struct_struct;

typedef struct {
        char            *name;
        unsigned long   sw_num;
        long            *sw_val;
        typecode        *type;
    }           tc union_member;

typedef struct {
        char            *name;
        typecode        *switchtype;
        unsigned long   memnum;
        tc_union_member *member;
    } tc union_struct;

typedef struct {
        char            *name;
        unsigned long   memnum;
        char            **member;
    } tc_enum_struct;

typedef struct {
        unsigned long   maxlength;
    } tc string_struct;
```

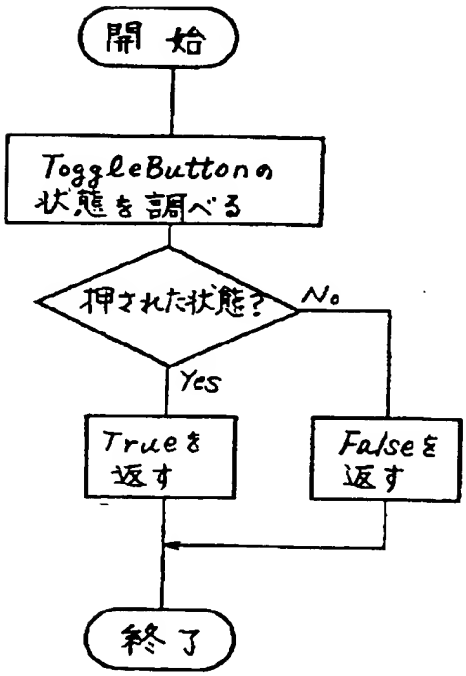
【図１９】



【図９】



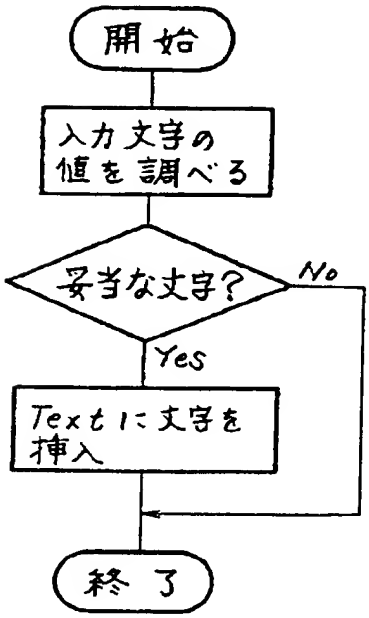
【図１０】



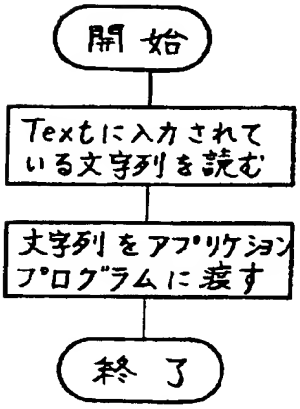
【図4】

```
#include "ui.isl"

enum 焼き具合型 { レア, ミディアム, ウェルダン };

enum サイズ型 { S, M, L };

struct ハンバーガー型 {
  焼き具合型 焼き具合;
  struct {
    boolean ケチャップ;
    boolean マスタード;
    boolean ピクルス;
    boolean オニオン;
    boolean マヨネーズ;
  } トッピング;
  unsigned long 個数;
};

struct フライ型 {
  サイズ型 サイズ;
  unsigned long 個数;
};

struct ドリンク型 {
  enum { アップルジュース, オレンジジュース, グレープジュース,
         コーラ, ジンジャエール, コーヒー } 種類;
  サイズ型 サイズ;
  unsigned long 個数;
};
```

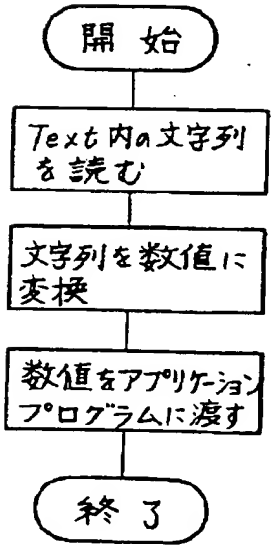
【図12】



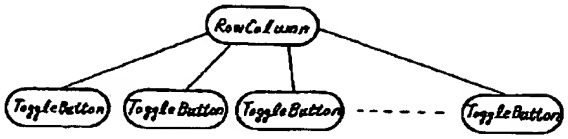
【図20】



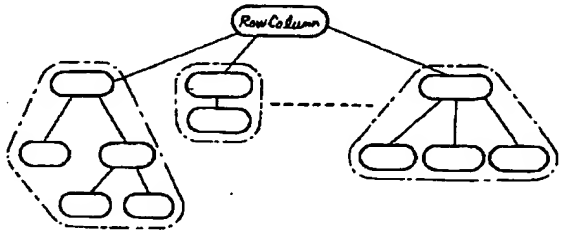
【図14】



【図16】



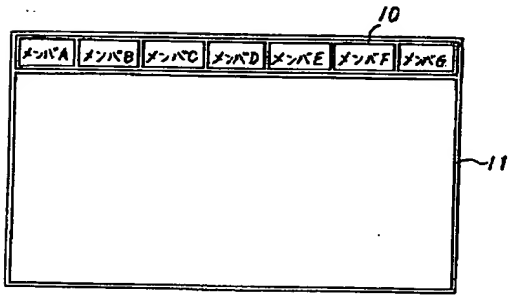
【図22】



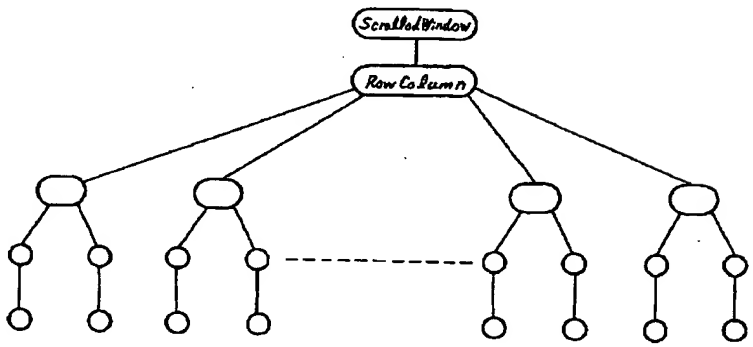
【図５】

```
UImanager uim;
UIdialog  root, メニュー欄;
UIoutput  メッセージ欄;
UIinput   ハンバーガー欄 フライ欄 ドリンク欄;
UIevent   注文ボタン, 取消ボタン;

string メッセージ;

void 注文処理()
{
  メッセージ欄.put("いらっしゃいませ。ご注文をどうぞ。");
}

void 取消処理()
{
  メッセージ欄.put("注文を取り消しました。");
}

void main()
{
  require uim;

  root = uim.dialog("ハンバーガー注文システム", 0, 1);
  メッセージ欄 = uim.output("", root, typeof(メッセージ), 1);
  メニュー欄 = uim.dialog("", root, 1);
  注文ボタン = uim.event("注文", メニュー欄, 注文処理, 1);
  取消ボタン = uim.event("取り消し", メニュー欄, 取消処理, 1);
  ハンバーガー欄 = uim.input("ハンバーガー", root, ハンバーガー型, 1);
  フライ欄 = uim.input("フライ", root, フライ型, 1);
  ドリンク欄 = uim.input("ドリンク", root, ドリンク型, 1);
  uim.activate();
}
```

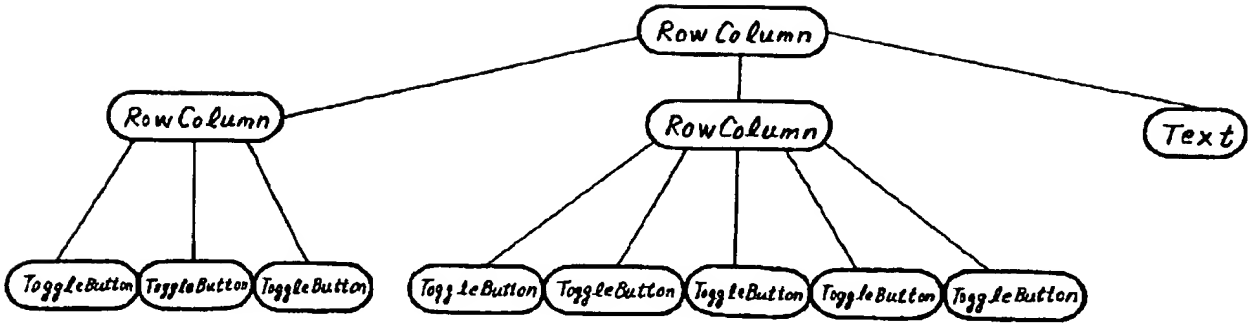
【図２７】



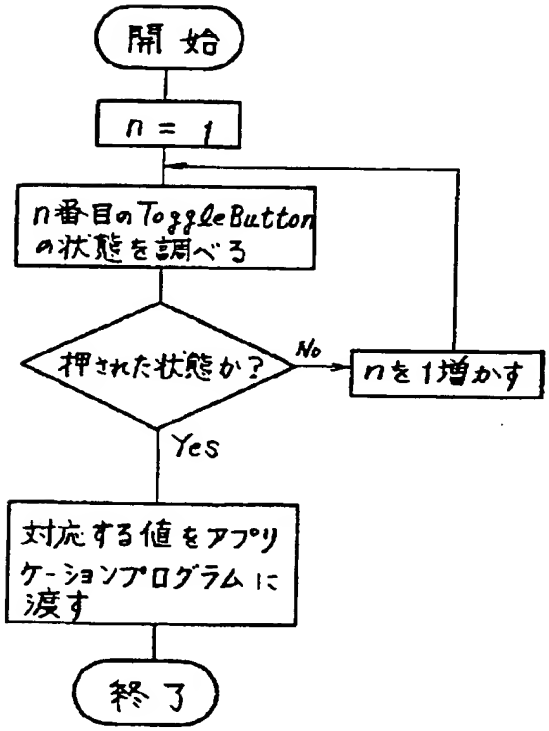
【図３４】



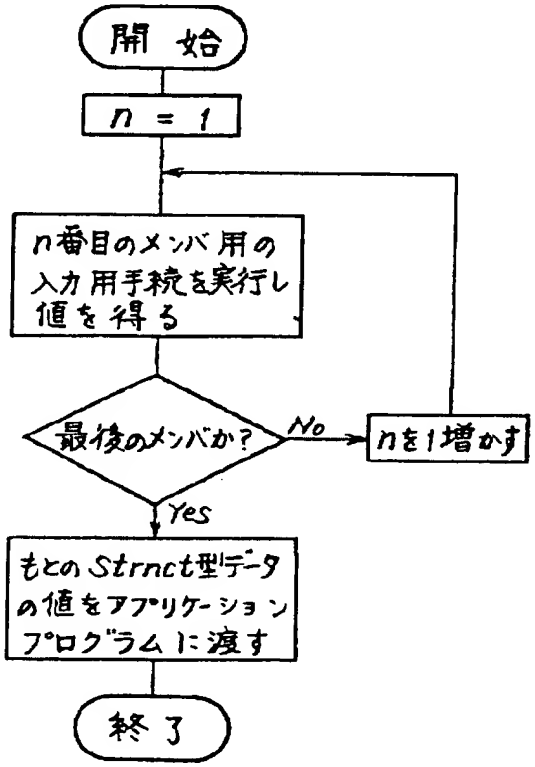
【図６】


RowColumn
RowColumn
RowColumn
Text
ToggleButton
ToggleButton
ToggleButton
ToggleButton
ToggleButton
ToggleButton
ToggleButton
ToggleButton

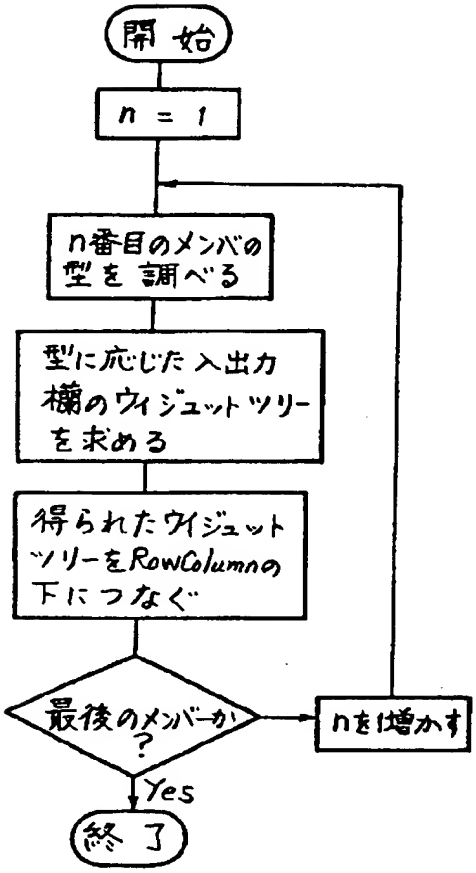
【図１８】


開始
n = 1
n番目のToggleButtonの状態を調べる
押された状態か？
No
nを1増かす
Yes
対応する値をアプリケーションプログラムに渡す
終了

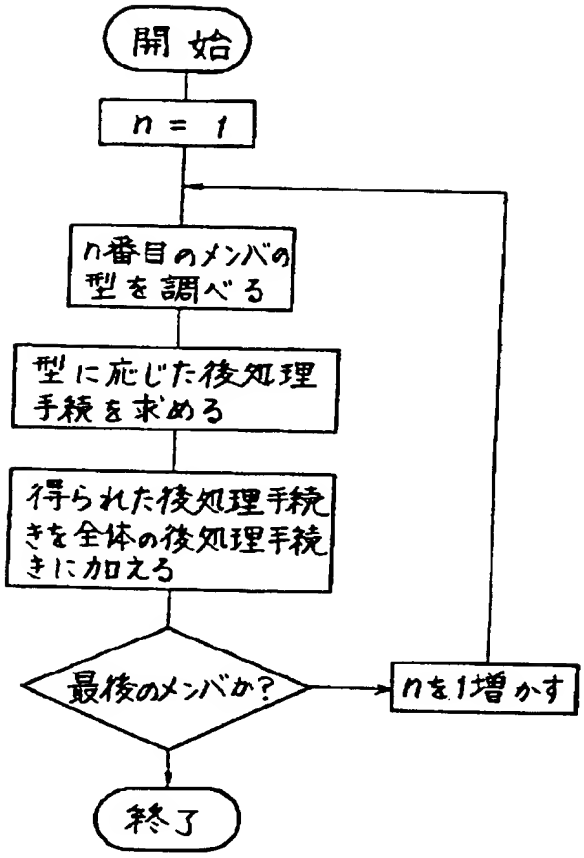
【図２５】


開始
n = 1
n番目のメンバ用の入力用手続を実行し値を得る
最後のメンバか？
No
nを1増かす
Yes
もとのStrnct型データの値をアプリケーションプログラムに渡す
終了

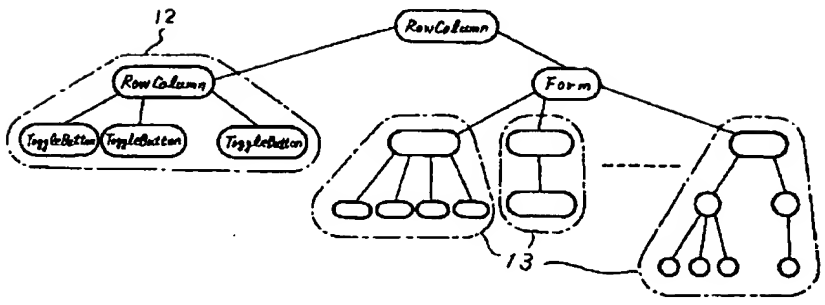
【図２３】


開始
n = 1
n番目のメンバの型を調べる
型に応じた入出力横のウィジットツリーを求める
得られたウイジットツリーをRowColumnの下につなぐ
最後のメンバーか?
Yes
nを1増かす
終了

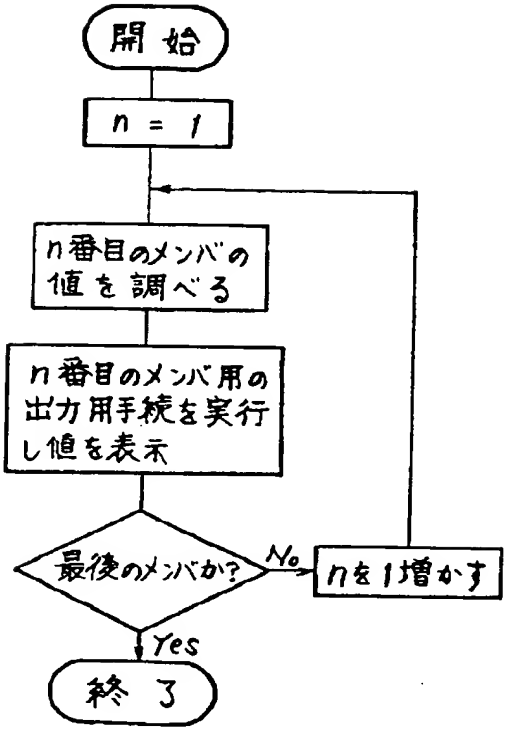
【図２４】


開始
n = 1
n番目のメンバの型を調べる
型に応じた後処理手続を求める
得られた後処理手続きを全体の後処理手続きに加える
最後のメンバか?
nを1増かす
終了

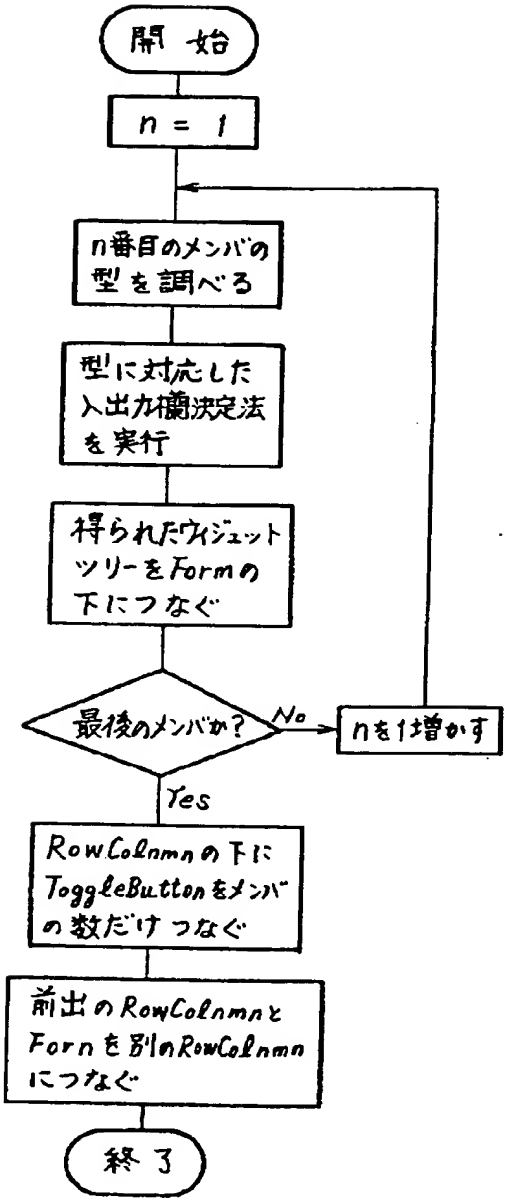
【図２８】


12
RowColumn
RowColumn
Form
ToggleButton
ToggleButton
ToggleButton
13

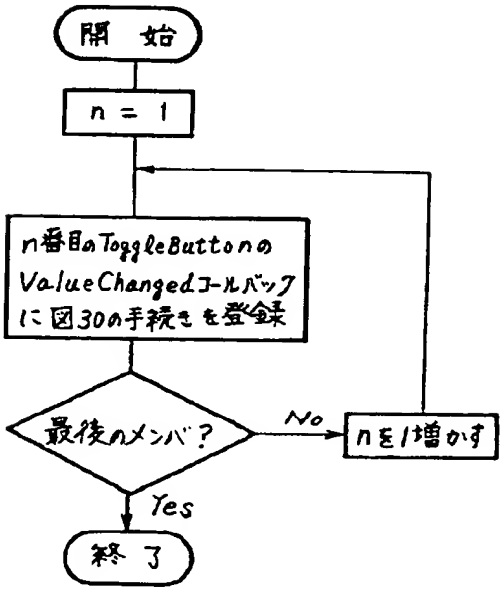
【図２６】


開始
n = 1
n番目のメンバの値を調べる
n番目のメンバ用の出力用手続を実行し値を表示
最後のメンバか?
No
nを1増かす
Yes
終了

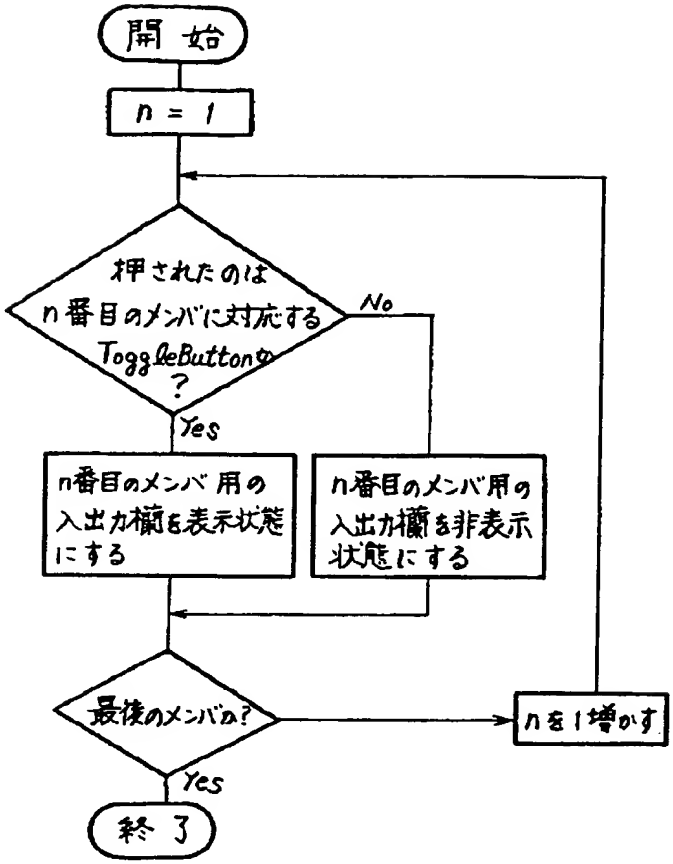
【図２９】


開始
n = 1
n番目のメンバの型を調べる
型に対応した入出力機決定法を実行
得られたウィジェットツリーをFormの下につなぐ
最後のメンバか?
No
nを1増かす
Yes
RowColumnの下にToggleButtonをメンバの数だけつなぐ
前出のRowColumnとFormを引のRowColumnにつなぐ
終了

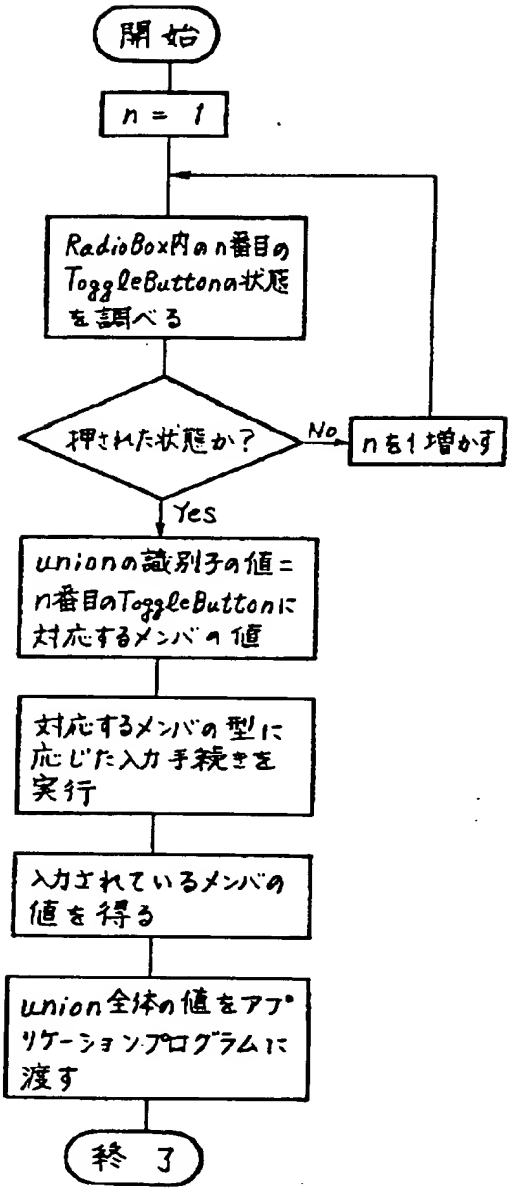
【図３１】


開始
n = 1
n番目のToggleButtonのValueChangedコールバックに図30の手続きを登録
最後のメンバ?
No
nを1増かす
Yes
終了

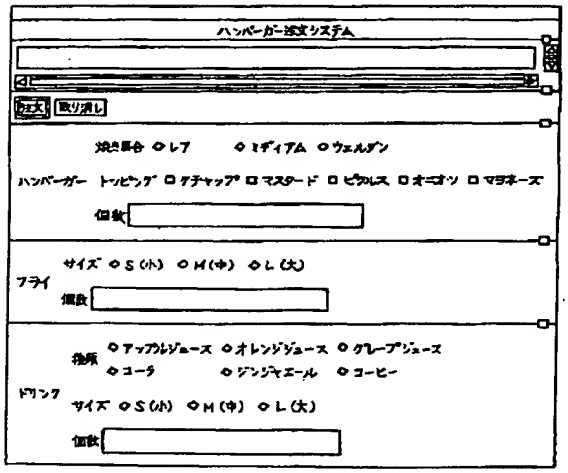
【図３０】


開始
n = 1
押されたのは
n番目のメンバに対応する
ToggleButtonか
?
No
Yes
n番目のメンバ用の
入出力欄を表示状態
にする
n番目のメンバ用の
入出力欄を非表示
状態にする
最後のメンバか?
n を1増かす
Yes
終了

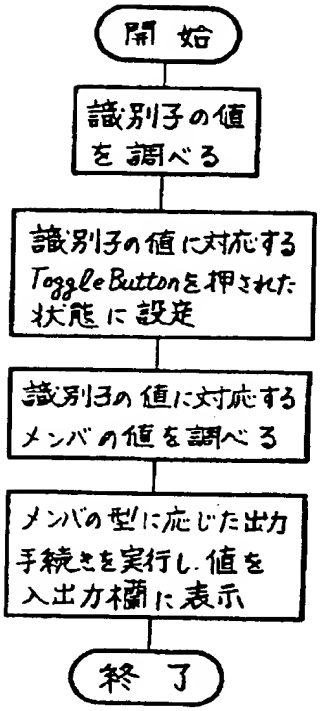
【図３２】


開始
n = 1
RadioBox内のn番目の
ToggleButtonの状態
を調べる
押された状態か?
No
n を1増かす
Yes
unionの識別子の値=
n番目のToggleButtonに
対応するメンバの値
対応するメンバの型に
応じた入力手続きを
実行
入力されているメンバの
値を得る
union全体の値をアプ
リケーションプログラムに
渡す
終了

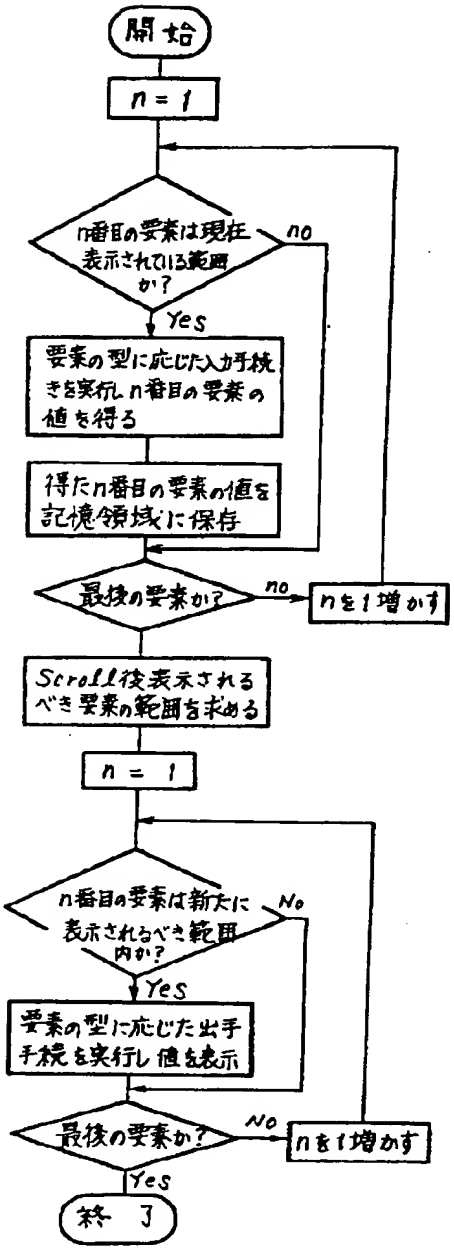
【図４１】


ハンバーガー注文システム
注文
取り消し
焼き具合 ○レア ○ミディアム ○ウェルダン
ハンバーガー トッピング □ケチャップ □マスタード □ピクルス □オニオン □マヨネーズ
個数
フライ
サイズ ○S(小) ○M(中) ○L(大)
個数
飲料 ○アップルジュース ○オレンジジュース ○グレープジュース
○コーラ ○ジンジャエール ○コーヒー
ドリンク
サイズ ○S(小) ○M(中) ○L(大)
個数

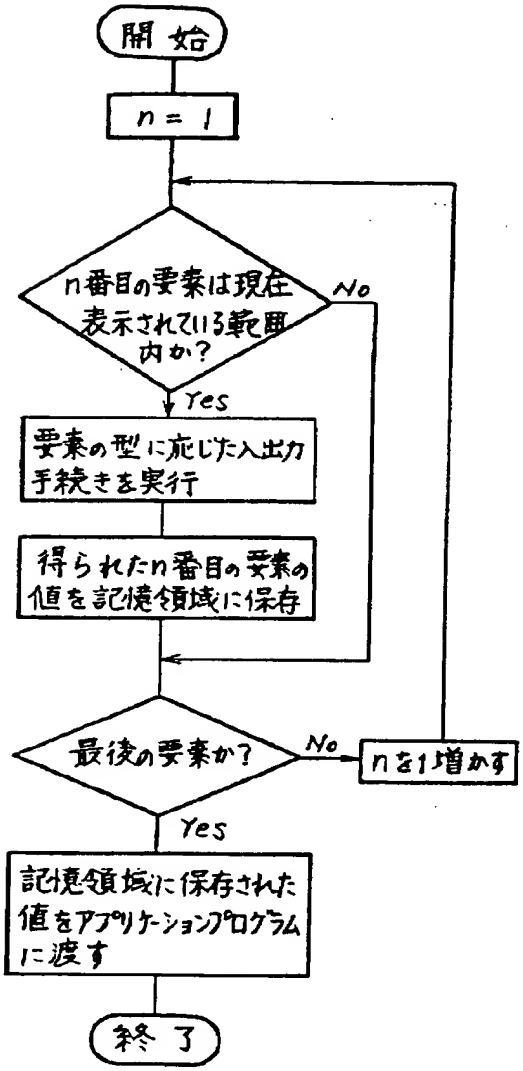
【図３３】


開始
識別子の値を調べる
識別子の値に対応するToggleButtonを押された状態に設定
識別子の値に対応するメンバの値を調べる
メンバの型に応じた出力手続きを実行し値を入出力欄に表示
終了

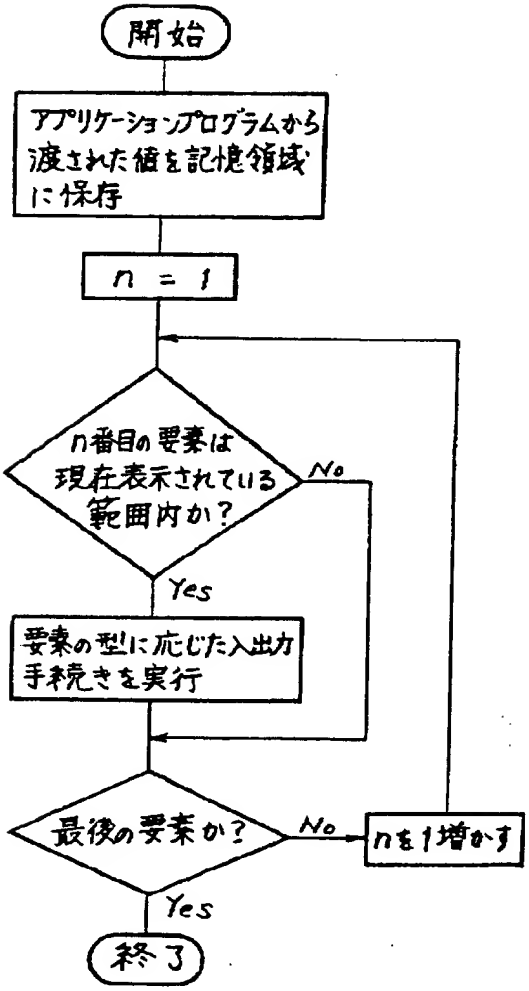
【図３５】


開始
n = 1
n番目の要素は現在表示されている範囲か?
no
Yes
要素の型に応じた入力手続きを実行しn番目の要素の値を得る
得たn番目の要素の値を記憶領域に保存
最後の要素か?
no
nを1増かす
Scroll後表示されるべき要素の範囲を求める
n = 1
n番目の要素は新たに表示されるべき範囲内か?
No
Yes
要素の型に応じた出手手続を実行し値を表示
最後の要素か?
No
nを1増かす
Yes
終了

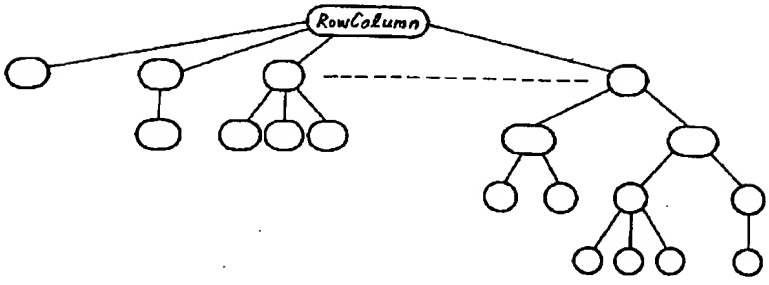
【図３６】


開始
n = 1
n番目の要素は現在表示されている範囲内か?
No
Yes
要素の型に応じた入出力手続きを実行
得られたn番目の要素の値を記憶領域に保存
最後の要素か?
No
nを1増かす
Yes
記憶領域に保存された値をアプリケーションプログラムに渡す
終了

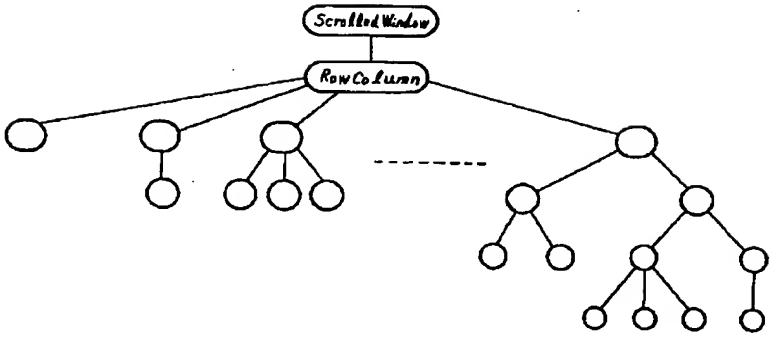
【図３７】


開始
アプリケーションプログラムから渡された値を記憶領域に保存
n = 1
n番目の要素は現在表示されている範囲内か?
No
Yes
要素の型に応じた入出力手続きを実行
最後の要素か?
No
nを1増かす
Yes
終了

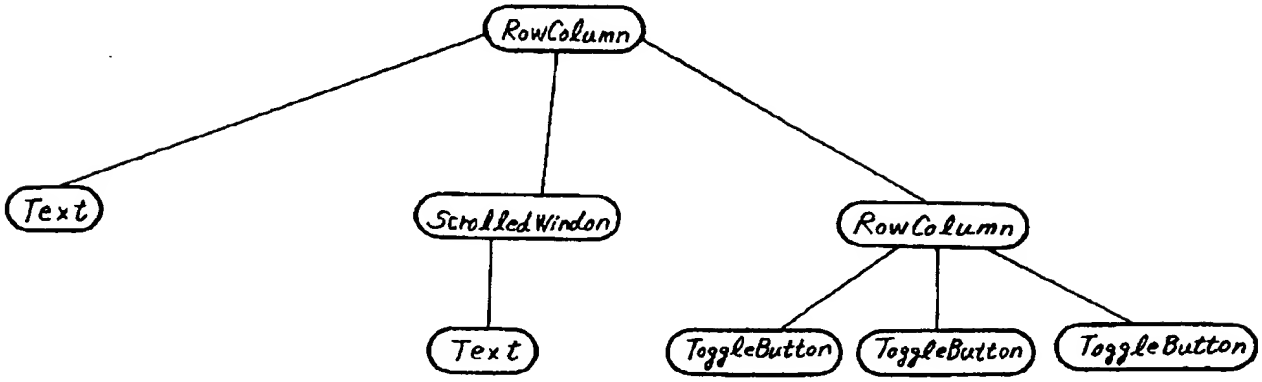
【図３８】


RowColumn

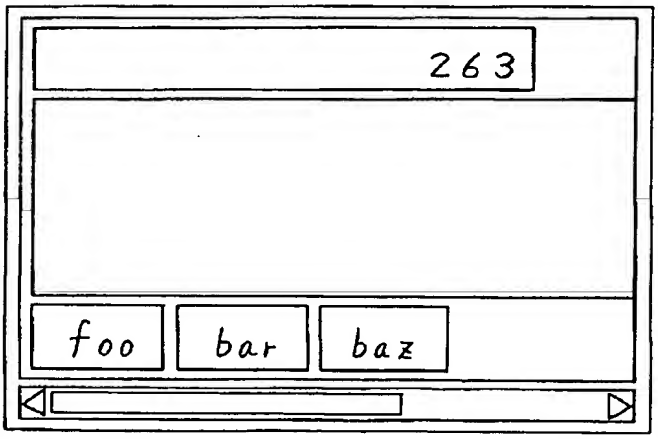
【図３９】


ScrolledWindow
RowColumn

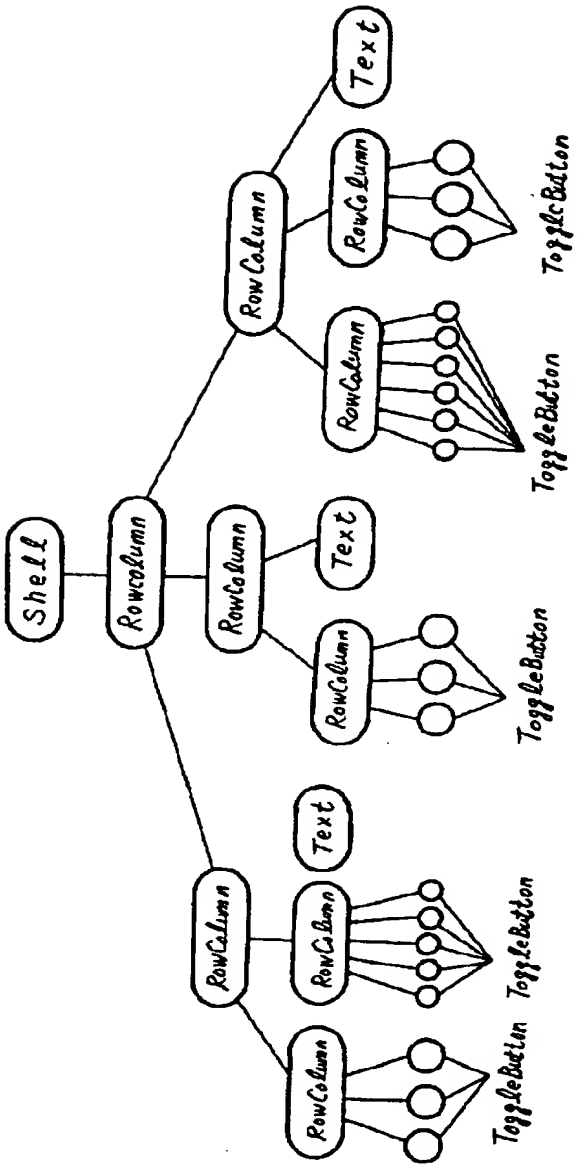
【図４２】


RowColumn
Text
ScrolledWindow
RowColumn
Text
ToggleButton
ToggleButton
ToggleButton

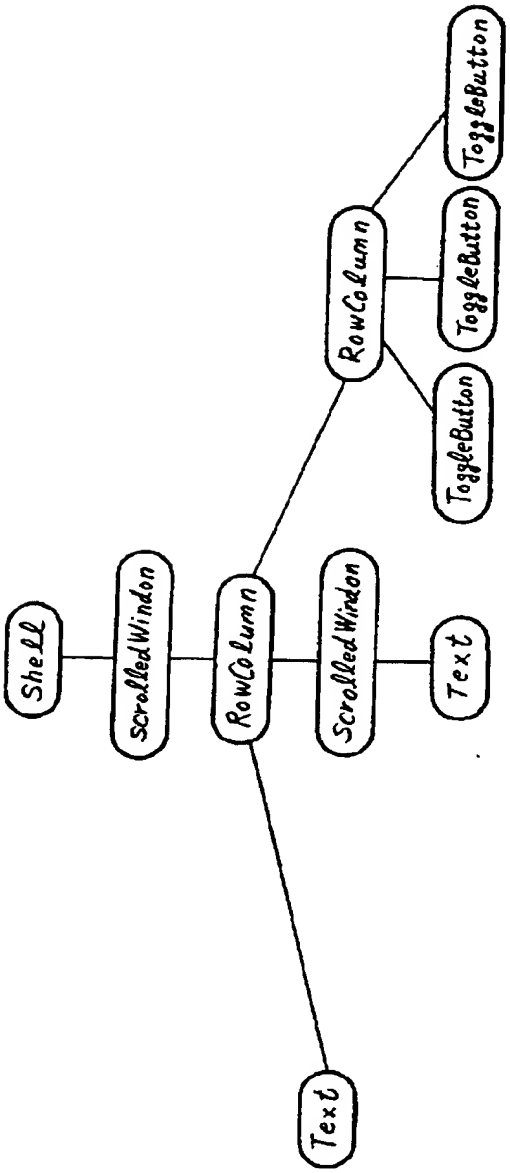
【図４４】


263
foo
bar
baz

【図４０】


Shell
RowColumn
RowColumn
RowColumn
RowColumn
RowColumn
RowColumn
RowColumn
Text
Text
Text
ToggleButton
ToggleButton
ToggleButton
ToggleButton
ToggleButton
ToggleButton

【図４３】


Shell
ScrolledWindow
RowColumn
ScrolledWindow
RowColumn
Text
Text
ToggleButton
ToggleButton
ToggleButton

【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成１３年２月１６日（２００１．２．１６）

【公開番号】特開平８－２６３２４３
【公開日】平成８年１０月１１日（１９９６．１０．１１）
【年通号数】公開特許公報８－２６３３
【出願番号】特願平７－９１９１１
【国際特許分類第７版】
G06F 3/14 310
【ＦＩ】
G06F 3/14 310 C

【手続補正書】
【提出日】平成１２年１月１４日（２０００．１．１４）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】発明の名称
【補正方法】変更
【補正内容】
【発明の名称】　データ入出力方法及び計算機装置
【手続補正２】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正内容】
【特許請求の範囲】
【請求項１】　ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る表示画面を計算機の表示装置に出現させ、この表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うデータ入出力方法において、
アプリケーションプログラム実行時に、アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を抽出し、
この抽出されたデータの型に基づき、予め記憶されたデータの型と表示すべき画面の構成との対応関係を参照して、前記アプリケーションプログラムに対応する画面の構成を求め、
この求められた構成に従って表示画面を前記表示装置に出現させ、
この出現させた表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うことを特徴とするデータ入出力方法。
【請求項２】　ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る画面を表示可能である表示手段と、
前記アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を抽出する抽出手段と、
データの型と前記表示手段に表示すべき画面の構成との対応関係を複数のデータの型について記憶する記憶手段と、
前記抽出手段の抽出結果と前記記憶手段に記憶された対応関係に従って表示画面を前記表示装置に表示させる表示制御手段と、
この表示画面を介して入力されるデータの前記アプリケーションプログラムへの送信及び前記アプリケーションプログラムから出力されるデータの前記表示画面を介した提示の少なくとも一方を行う入出力制御手段と、
この入出力制御手段により送信されたデータを用いた前記アプリケーションプログラムの実行及び入出力制御手段に対する前記アプリケーションプログラムの実行結果の出力の少なくとも一方を行う実行手段とを備えたことを特徴とする計算機装置。
【手続補正３】
【補正対象書類名】明細書
【補正対象項目名】０００７
【補正方法】変更
【補正内容】
【０００７】
【課題を解決するための手段】本発明（第１の発明）は、ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る表示画面を計算機の表示装置に出現させ、この表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うデータ入出力方法において、アプリケーションプログラム実行時に、アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を抽出し、この抽出されたデータの型に基づき、予め記憶されたデータの型と表示すべき画面の構成との対応関係を参照して、前記アプリケーションプログラムに対応する画面の構成を求め、

この求められた構成に従って表示画面を前記表示装置に出現させ、この出現させた表示画面を介してアプリケーションプログラムとユーザとの間のデータの入出力を行うことを特徴とする。

【手続補正４】
【補正対象書類名】明細書
【補正対象項目名】０００８
【補正方法】変更
【補正内容】
【０００８】本発明（第２の発明）に係る計算機装置は、ユーザからアプリケーションプログラムに対するデータの入力あるいはアプリケーションプログラムからのデータの出力を司る画面を表示可能である表示手段と、前記アプリケーションプログラムが入力及び出力の少なくとも一方を行うデータの型の定義を抽出する抽出手段と、データの型と前記表示手段に表示すべき画面の構成との対応関係を複数のデータの型について記憶する記憶手段と、前記抽出手段の抽出結果と前記記憶手段に記憶された対応関係に従って表示画面を前記表示装置に表示させる表示制御手段と、この表示画面を介して入力されるデータの前記アプリケーションプログラムへの送信及び前記アプリケーションプログラムから出力されるデータの前記表示画面を介した提示の少なくとも一方を行う入出力制御手段と、この入出力制御手段により送信されたデータを用いた前記アプリケーションプログラムの実行及び入出力制御手段に対する前記アプリケーションプログラムの実行結果の出力の少なくとも一方を行う実行手段とを備えたことを特徴とする。

【手続補正５】
【補正対象書類名】明細書
【補正対象項目名】０００９
【補正方法】削除